C++ ASSIGNMENT

1.Ques :Given the head of a sorted linked list, delete all nodes that have duplicate numbers,
leaving only distinct numbers from the original list. Return the linked list sorted as well.
[Leetcode 82].

Ans:
```
Class solution{
    public:
        ListNode* deleteDuplicates(ListNode* head) {
       ListNode *dummy=new ListNode(0,head);
      ListNode *prev=dummy;

       while(head!=NULL){

       if(head->next!=NULL &&
    head->val==head->next->val){

                       while(head->next!=NULL &&
        head->val==head->next->val)head=head->next;


             prev->next=head->next;
          }

          else prev=prev->next;


          head=head->next;
```

```
            }

            return dummy->next;

          }
       };
```

2.Ques :Given the head of a singly linked list, sort the list using insertion sort, and return the sorted
list's head. [Leetcode 147].

Ans:
```
class Solution {
public:
   ListNode* insertionSortList(ListNode* head) {
      ListNode* dummy=new ListNode(10);
      while(head){
         ListNode* next=head->next;
         ListNode* temp=dummy;
         while(temp->next && temp->next->val<head->val){
            temp=temp->next;
         }
         head->next=temp->next;
         temp->next=head;
         head=next;
      }
      return dummy->next;
   }
};
```

3.Ques :Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.The first node is considered odd, and the second node is even, and so on.
Note that the relative order inside both the even and odd groups should remain as it was in
the input.
You must solve the problem in O(1) extra space complexity and O(n) time complexity.
[Leetcode 328]

Ans:
```
class Solution {
public:
  ListNode* oddEvenList(ListNode* head) {
    if(head==NULL || head->next==NULL) return head;
    ListNode* even=head->next;
    ListNode* odd=head;
    ListNode* evenHead=even;
    while(even && even->next){
      odd->next=odd->next->next;
      odd=odd->next;
      even->next=even->next->next;
      even=even->next;
    }
    odd->next=evenHead;
    return head;
  }
};
```

4.Ques : You are given two non-empty linked lists representing

two non-negative integers. The
digits are stored in reverse order, and each of their nodes
contains a single digit. Add the
two numbers and return the sum as a linked list. [Leetcode 2]

Ans: class Solution {

```
public:
   ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
      ListNode* a=l1;
      ListNode* b=l2;
      ListNode* c=new ListNode(10);
      ListNode* temp=c;
      int carry=0;
      while(a || b){
         int sum=0+carry;
         if(a){
            sum+=a->val;
            a=a->next;
         }
         if(b){
            sum+=b->val;
            b=b->next;
         }
         carry=sum/10;
         sum=sum%10;
         temp->next=new ListNode(sum);
         temp=temp->next;
      }
      if(carry==1) temp->next=new ListNode(1);
      return c->next;
   }
```

};

5.Ques :Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.
Return the decimal value of the number in the linked list.
The most significant bit is at the head of the linked list. [Leetcode 1290]

Ans:
```
class Solution {
public:
   int getDecimalValue(ListNode* head) {
      ListNode* temp=head;
      int res=0;
      while(temp){
         res*=2;
         res+=temp->val;
         temp=temp->next;
      }
      return res;
   }
};
```

6.Ques :In a linked list of size n, where n is even, the ith node (0-indexed) of the linked list is known as the twin of the (n-1-i)th node, if 0 <= i <= (n / 2) - 1.
For example, if n = 4, then node 0 is the twin of node 3, and node 1 is the twin of node 2. These are the only nodes with twins for n = 4.The twin sum is defined as the sum of a node and its twin.
Given the head of a linked list with even length, return the maximum twin sum of the linked
list. [Leetcode 2130]

Ans:

```
class Solution {
public:
   int pairSum(ListNode* head) {
      ListNode* temp=head;
      vector<int> ans;
      int len=0;
      while(temp){
         len++;
         ans.push_back(temp->val);
         temp=temp->next;
      }
      int i=0;
      int j=len-1;
      int maxsum=INT_MIN;
      while(i<=j){
         int sum=ans[i]+ans[j];
         i++;
         j--;
         maxsum=max(maxsum,sum);
      }
      return maxsum;
   }
};
```

7.Ques:Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list. k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is. You may not alter the values in the list's nodes, only nodes themselves may be changed. [Leetcode 25]

Ans:
```
class Solution {
public:
   ListNode* reverseList(ListNode* head) {
      ListNode* curr=head;
      ListNode* prev=NULL,*Next=NULL;
      while(curr!=NULL){
         Next=curr->next;
         curr->next=prev;
         prev=curr;
         curr=Next;
      }
      return prev;
   }
   ListNode* getKthNode(ListNode* temp,int k){
      while(temp && k>1){
         temp=temp->next;
         k--;
      }
      return temp;
   }
   ListNode* reverseKGroup(ListNode* head, int k) {
      ListNode* temp=head;
      ListNode* prev=NULL;
      while(temp){
         ListNode* kthNode=getKthNode(temp,k);
         if(kthNode==NULL){
            if(prev) prev->next=temp;
            break;
         }
         ListNode* Next=kthNode->next;
```

```
            kthNode->next=NULL;
            reverseList(temp);
            if(temp==head) head=kthNode;
            else prev->next=kthNode;
            prev=temp;
            temp=Next;
        }
        return head;
    }
};
```